Amendments to the Specification:

Please replace the paragraph beginning at page 1, line 21 with the following amended paragraph:

A receiving application associated with a receiving IP stack receives and processes the information. Each layer of the receiving IP stack performs various communication functions and format conversions in reverse going from the physical layer, the data link layer, the network layer, the transport layer, and then to the receiving application. In [[the]] a conventional network, applications send and receive messages from each other and use the IP stack as a conduit for data. Notwithstanding these messages, other information being transmitted between the sending and receiving IP stacks is not typically made available to either the sending or receiving applications.

Please replace the paragraph beginning at page 1, line 29 with the following amended paragraph:

While layered protocols such as used in [[the]] a conventional IP stack have some advantages, they are [[have been]] typically obtained by lowering programmatic flexibility. For example, application data is encapsulated with protocol-generated headers whose content cannot be accessed and controlled by the application itself. Applications are masked from the inner operation of a network protocol and network operation. This inflexibility makes it difficult for an application to send data encapsulated with a non-standard header when required or monitor operation of the network.

Please replace the paragraph beginning at page 3, line 4 with the following amended paragraph:

FIG. 1 is a block diagram illustrating a network 100 using an inner layer application programming interface (IL API) to communicate between nodes on [[a]] network 100. Network 100 includes a transmit application 102 with a corresponding TCP/IP stack 104, a data link layer

106 and [[a]] an inner layer application programming interface (IL API) 108 facilitating communication between transmit application 102 and layers within TCP/IP stack 104. Further, network 100 also includes a first intermediate gateway or router node represented by IP stack 110 and data link layer 112 and a second intermediate gateway or router node represented by IP stack 114 and data link layer 116. Receive application 118 in network 100 has a TCP/IP stack 120, data link layer 122 and [[a]] an IL API 124. Physical connection 126 provides connection to each of these nodes through their respective data link layers using a physical access protocol such as Carrier Sense Multiple Access/Collision Detect (CSMA/CD).

Please replace the paragraph beginning at page 3, line 15 with the following amended paragraph:

Conventional layered communications provides applications with [[application to application]] application-to-application or peer-to-peer [[or]] communication capabilities. Information at the lower layers of the protocol stack are masked from the application through abstract interfaces. This simplifies network programming over the IP stack but does not provide much flexibility if access to these other layers is desired. IL API 108 and IL API 124 provides this communication capability to both transmit application 102 and receive application 118. For example, transmit application 102 and receive application 118 have access to IP stack 110 and IP stack 114 directly using their respective [[IP]] IL API. Additionally, transmit application 102 and receive application 118 also have access to other protocol layers, [[using the IP API]] such as data link layer 112 and data link layer 116, using their respective IL API.

Please replace the paragraph beginning at page 3, line 25 with the following amended paragraph:

FIG. 2 illustrates many different types of network information available at these different layers in the protocol stack. This block diagram illustrates an application 202 passing through an IL API 204 to gain access to a transport layer 206, a network layer 208, and a data link layer 210. At transport layer 206, application 202 has access to the transport protocols TCP 212, UDP 214, and other transport 216. TCP 212 or Transmission Control Protocol is a connection-oriented protocol that provides a reliable, full-duplex, byte stream for a user process. Most conventional Internet applications use TCP 212 and allow TCP 212 to interface with the IP layers below. UDP 214 [[212]] or User Datagram Protocol is a connectionless protocol also for user processes, however, it does not guarantee that UDP datagrams will ever reach their intended destination. Because TCP and UDP both access the IP layer the protocol is often referred to as simply TCP/IP.

Please replace the paragraph beginning at page 4, line 4 with the following amended paragraph:

Network layer 208 provides application 202 with access through IL API 204 to information carried over Appletalk 218, IPv4 220, IPv6 222, and IPX 224. These protocols provide packet delivery services and routing capabilities for transport protocols such as TCP 212 and UDP 214. Networks based on Appletalk 218 and IPX 224 can be integrated to work with the TCP and UDP transport protocols. In addition, routers, switches, hubs and other network devices exchange status and network routing information describing network layer resources using ICMP (Internet Control Message Protocol) and IGMP (Internet Gateway Message Protocol). Appletalk 218 provides packet delivery services primarily to computers designed by Apple Computer of Cupertino, California. IPv4 220 (version 4) provides 32-bit addresses and IPv6 222 (version 6) provides 64-bit addresses in the Internet Protocol (IP) defined in specification DOD-STD-1777. Further references to the IP protocol includes these additional protocols described above.

Please replace the paragraph beginning at page 4, line 16 with the following amended paragraph:

Application 202 also has access to data link layer 210 through IL API 204. Fiber distributed data interface (FDDI) protocol 226 is a standard for data transmission on fiber optic lines in a local area network. FDDI protocol 226 [[protocol]] is based on the token ring protocol and in addition can support thousands of users. In addition, application 202 can also access information from Ethernet 228 through IL API 204. Ethernet 228 is a widely-installed local area network technology and specifies sharing physical access over coaxial cable or special grades of twisted pair wires providing a wide range of transmission speeds. Devices connected to an Ethernet network generally compete for access to the physical medium using the CSMA/CD protocol.

Please replace the paragraph beginning at page 4, line 25 with the following amended paragraph:

FIG. 3 is a block diagram illustrating how the IL API works to provide access to a Internet Protocol (IP) stack 300. IP stack 300 includes application 302, transport layer 304, network layer 306, data link layer 308 each connected to IL API 312. In one implementation, layers in IP stack 300 [[product]] <u>produce</u> an Ethernet packet 310 with a data payload and headers from each of the various layers.

Please replace the paragraph beginning at page 4, line 30 with the following amended paragraph:

In conventional network communication, application 314 and application 316 communicate directly with a network protocol with either TCP 318 or UDP 320 for a connection or connectionless type communication. As an alternative, both application 314 and application 316 can communicate with transport layer <u>304</u> [[38]] through TCP Socket 334 in IL API 312. Using IL API 312 to access the transport layer as well as other inner layers of the network protocol provides a more uniform interface to the stack.

Please replace the paragraph beginning at page 5, line 4 with the following amended paragraph:

Application 314 and application 316 can use IL API 312 to access network layer 306 and data link layer 308 in ways previously unavailable. For example, application 314 can access

Internet Control Message Protocol (ICMP)/Internet Group Multicast Protocol (IGMP) 324 resources and interact with routers, switches, hubs, gateways, and hosts communicating with each other about errors and system control. ICMP provides message control and error-reporting protocol between a host server and a gateway to the Internet. ICMP uses Internet Protocol (IP) datagrams that IL API 312 provides to an application. On conventional systems, this information is processed by the TCP/IP protocol and is not available directly to the application. IGMP is used to support multicasting between nodes on a network and provides information to applications through IL API 312 in a similar manner. Application 314 also has access to ARP 326 and RARP 328 resources. Application 314 opens a socket using IP Socket 336 interface and establishes a direct connection with network layer 306. Because application 314 bypasses transport layer 318, ARP 326 and RARP 328 ~~information is~~ resources are exposed and available for application 314 to process. For example, ARP 326 ~~information includes~~ resources include Media Access Control (MAC) addresses associated with each Ethernet device on a network.

Please replace the paragraph beginning at page 5, line 20 with the following amended paragraph:

Application 314 operates in a similar manner with respect to data link layer 308. To gain access to data link layer 308, application 314 establishes a session directly to data link layer 308 through link socket 338. Once the session through link socket 338 is created, [[the]] application 314 has access to information in data link layer 330 and physical layer 332. For example, application 314 can create customized headers for an Ethernet packet 310 creating TCP Header and IP Header as illustrated in Ethernet packet 310 in FIG. 3. Ethernet header and Ethernet trailer are added by an Ethernet type data link layer 330. This provides an application with additional flexibility when developing network management software or developing other routines that need access to lower layers of the network protocol stack.

Please replace the paragraph beginning at page 5, line 29 with the following amended paragraph:

FIG. 4. is a block diagram depicting a computer system 400 that provides the IL API and IP stack to applications. Computer system includes a memory 402, a processor 404, a network communication port 406, a secondary storage 408, and input-output ports 410. Processor can be a general purpose processor such as manufactured by Intel ~~Incorporated~~ Corporation of Santa Clara, California or can be a specialized ASIC or other type of processor device. Network communication port 406 can be implemented as a Ethernet card or built-in communication port on a computer and secondary storage 408 is a hard-disk, CDROM, or other mass storage device. Input-output ports 410 includes ports for corresponding peripheral devices such as keyboard, mouse, printer, display, and scanner.

Please replace the paragraph beginning at page 6, line 6 with the following amended paragraph:

Memory 402 includes an application 414, an inner layer API (IL API) 416, inner layer extensions[[,]] 418, virtual machine runtime environment 420, TCP/IP protocol 422, network resources 423 and operating system 424. Application 414 is an application that can access one or more different layers of a network protocol stack such as TCP/IP protocol 422. Generally, application 414 should be a user application but may need to be run with increased permissions such as "root" or "superuser" due to the sensitive information accessible within the inner layers of TCP/IP protocol 422.

Please replace the paragraph beginning at page 6, line 13 with the following amended paragraph:

Inner layer API 416 is the interface routines linked into application 414 that provides direct access to the transport, the network, data link layers and physical layers in the protocol stack. Inner layer extensions 418 include any supporting routines necessary to make the IL API 416 [[312]] available on the given platform. In some cases, this could involve recompiling an operating system kernel to include these particular functionalities not previously available to applications. In an object-oriented implementation, such as using the Java programming language by Sun Microsystems of Mountain View, California, these extensions can be dynamically loaded at run-time or immediately when they are loaded into the overall system. Because Java allows dynamic loading of routines, inner layer extensions 418 can be loaded as application 414 requires.

Please replace the paragraph beginning at page 7, line 16 with the following amended paragraph:

Given several layers to communicate with, application selects a network layer to establish communication (504). In part, the layer selected depends on the type of datagram the application has created. If the application creates a transport session using a transport socket such as TCP 334 in FIG. 3, the application provides the data and necessary headers. However, a network session uses a network socket such as IP Socket 336 [[446]] in FIG. 3 and the application needs to create the appropriate network layer TCP header or UDP header around the data or payload section of each packet. Similarly, if the application creates a link layer session using link socket 338 then the application must also include IP header information in the packet.

Please replace the paragraph beginning at page 7, line 25 with the following amended paragraph:

The application also selects a layer in the network protocol stack depending on the layer a resource associated with the network device uses for communication. For example, the ICMP and IGMP tables are resources that use the IP protocol because they communicate [[that]] at the network layer in the protocol stack. Similarly, an ARP table is a resource that uses the link layer to communicate information about the network device, in particular an Ethernet or MAC address of the network device.

Please replace the paragraph beginning at page 8, line 12 with the following amended paragraph:

FIG. 6 is a flow-diagram depicting the operations used to access information about a resource associated with a network device. Initially, a network device receives a request from an application for information about a resource associated with a network device (602). The resource can be a routing table, a table with ICMP or IGMP information, a table of physical ports on a network device, an ARP table of MAC addresses or any other resource associated with a network device. The request typically [[typical]] selects a layer in a network protocol stack for communicating with the requested resource on the network device (604). The application then establishes an inner layer socket for communicating at the selected layer using an inner layer application programming interface (IL API) and a socket identifier associated with the requested resource (606). The inner layer socket communicates using the selected layer and bypasses other layers in the network protocol stack. Once established, the inner layer socket transmits the request for information about the resource through the inner layer socket and the socket identifier (608). The IL API receives the information about the resource from the network device in response to the transmission made through the inner layer socket (610) and passes the information about the resource through the inner layer socket to the application making the request.

Please replace the paragraph beginning at page 9, line 29 with the following amended paragraph:

While specific implementations have been described herein for purposes of illustration, various modifications may be made without departing from the spirit and scope of the invention. For example, implementations and examples are provided with reference to TCP/IP however, an alternate implementation could also be adapted to work with the Open Systems Interconnection [[(OSI)network]] (OSI) network model. In the OSI communication model, IP is in layer 3, and other layers are as illustrated in FIG. 3. Inner sockets for the transport, network and data link layer are described but an inner socket for a physical layer could also be implemented. The physical layer would provide information about the ports on a network device and information about the physical media being used. Additional implementations could be created using conventional procedural programming languages such as "C" as well as object-oriented programming environments/languages such as Java or C++. Furthermore, although aspects of the present invention are described as being stored in memory and other storage mediums, one skilled in the art will appreciate that these aspects can also be stored on or read from other types of computer-readable media, such as secondary storage devices, like hard disks, floppy disks, or CD-ROM; a carrier wave from the Internet; or other forms of RAM or ROM. Accordingly, the invention is not limited to the above-described embodiments, but instead is defined by the appended claims in light of their full scope of equivalents.